Finite difference methods for diffusion processes

Hans Petter Langtangen[1,2]

[1]Center for Biomedical Computing, Simula Research Laboratory
[2]Department of Informatics, University of Oslo

Aug 18, 2014

Note: **VERY PRELIMINARY VERSION**

Contents

1 The 1D diffusion equation

indexdiffusion equation, 1D indexheat equation, 1D

The famous *diffusion equation*, also known as the *heat equation*, rea

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2},$$

where $u(x,t)$ is the unknown function to be solved for, x is a coord space, and t is time. The coefficient α is the *diffusion coefficient* and det how fast u changes in time. A quick short form for the diffusion equ $u_t = \alpha u_{xx}$.

Compared to the wave equation, $u_{tt} = c^2 u_{xx}$, which looks very simi the diffusion equation features solutions that are very different from thos wave equation. Also, the diffusion equation makes quite different dem the numerical methods.

Typical diffusion problems may experience rapid change in the very be but then the evolution of u becomes slower and slower. The solution is very smooth, and after some time, one cannot recognize the initial sha This is in sharp contrast to solutions of the wave equation where the initi is preserved - the solution is basically a moving initial condition. The s wave equation $u_{tt} = c^2 u_{xx}$ has solutions that propagates with speed c without changing shape, while the diffusion equation converges to a *st solution* $\bar{u}(x)$ as $t \to \infty$. In this limit, $u_t = 0$, and $\bar{u}$ is governed by $\bar{u}''$ This stationary limit of the diffusion equation is called the *Laplace* equat arises in a very wide range of applications throughout the sciences.

It is possible to solve for $u(x,t)$ using a explicit scheme, but the ti restrictions soon become much less favorable than for an explicit scheme wave equation. And of more importance, since the solution u of the d equation is very smooth and changes slowly, small time steps are not co and not required by accuracy as the diffusion process converges to a st state.

1.1 The initial-boundary value problem for 1D diffu

To obtain a unique solution of the diffusion equation, or equivalently, t numerical methods, we need initial and boundary conditions. The d equation goes with one initial condition $u(x,0) = I(x)$, where I is a pr function. One boundary condition is required at each point on the ary, which in 1D means that u must be known, u_x must be known, combination of them.

We shall start with the simplest boundary condition: $u = 0$. The c initial-boundary value diffusion problem in one space dimension can specified as

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}, \quad x \in (0, L),\ t \in (0, T] \tag{1}$$

$$u(x, 0) = I(x), \qquad x \in [0, L] \tag{2}$$

$$u(0, t) = 0, \qquad t > 0, \tag{3}$$

$$u(L, t) = 0, \qquad t > 0\,. \tag{4}$$

quation (1) is known as a one-dimensional *diffusion equation*, also often referred o as a *heat equation*. With only a first-order derivative in time, only one *initial ondition* is needed, while the second-order derivative in space leads to a demand or two *boundary conditions*. The parameter α must be given and is referred to s the *diffusion coefficient*.

Diffusion equations like (1) have a wide range of applications throughout hysical, biological, and financial sciences. One of the most common applications propagation of heat, where $u(x, t)$ represents the temperature of some substance t point x and time t. Section **??** goes into several widely occurring applications.

.2 Forward Euler scheme

he first step in the discretization procedure is to replace the domain $[0, L] \times [0, T]$ y a set of mesh points. Here we apply equally spaced mesh points

$$x_i = i\Delta x, \quad i = 0, \ldots, N_x,$$

nd

$$t_n = n\Delta t, \quad n = 0, \ldots, N_t\,.$$

Ioreover, u_i^n denotes the mesh function that approximates $u(x_i, t_n)$ for $i = , \ldots, N_x$ and $n = 0, \ldots, N_t$. Requiring the PDE (1) to be fulfilled at a mesh oint (x_i, t_n) leads to the equation

$$\frac{\partial}{\partial t} u(x_i, t_n) = \alpha \frac{\partial^2}{\partial x^2} u(x_i, t_n), \tag{5}$$

he next step is to replace the derivatives by finite difference approximations. he computationally simplest method arises from using a forward difference in me and a central difference in space:

$$[D_t^+ u = \alpha D_x D_x u]_i^n\,. \tag{6}$$

Vritten out,

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}\,. \tag{7}$$

Ve have turned the PDE into algebraic equations, also often called discrete quations. The key property of the equations is that they are algebraic, which

makes them easy to solve. As usual, we anticipate that u_i^n is already co such that u_i^{n+1} is the only unknown in (7). Solving with respect to this u is easy:

$$u_i^{n+1} = u_i^n + \alpha \frac{\Delta t}{\Delta x^2} \left(u_{i+1}^n - 2u_i^n + u_{i-1}^n \right)\,.$$

The computational algorithm then becomes

1. compute $u_i^0 = I(x_i)$for $i = 0, \ldots, N_x$
2. for $n = 0, 1, \ldots, N_t$:
   (a) apply (8) for all the internal spatial points $i = 1, \ldots, N_x - 1$
   (b) set the boundary values $u_i^{n+1} = 0$ for $i = 0$ and $i = N_x$

The algorithm is compactly fully specified in Python:

```
x = linspace(0, L, Nx+1)    # mesh points in space
dx = x[1] - x[0]
t = linspace(0, T, Nt+1)    # mesh points in time
dt = t[1] - t[0]
Fo = a*dt/dx**2
u   = zeros(Nx+1)
u_1 = zeros(Nx+1)

# Set initial condition u(x,0) = I(x)
for i in range(0, Nx+1):
    u_1[i] = I(x[i])

for n in range(0, Nt):
    # Compute u at inner mesh points
    for i in range(1, Nx):
        u[i] = u_1[i] + Fo*(u_1[i-1] - 2*u_1[i] + u_1[i+1])

    # Insert boundary conditions
    u[0] = 0;  u[Nx] = 0

    # Update u_1 before next step
    u_1[:]= u
```

1.3 Backward Euler scheme

We now apply a backward difference in time in (5), but the same central di in space:

$$[D_t^- u = D_x D_x u]_i^n,$$

which written out reads

$$\frac{u_i^n - u_i^{n-1}}{\Delta t} = \alpha \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}\,.$$

ow we assume u_i^{n-1} is computed, but all quantities at the "new" time level n re unknown. This time it is not possible to solve with respect to u_i^n because this alue couples to its neighbors in space, u_{i-1}^n and u_{i+1}^n, which are also unknown. et us examine this fact for the case when $N_x = 3$. Equation (10) written for $= 1, \ldots, Nx - 1 = 1, 2$ becomes

$$\frac{u_1^n - u_1^{n-1}}{\Delta t} = \alpha \frac{u_2^n - 2u_1^n + u_0^n}{\Delta x^2} \tag{11}$$

$$\frac{u_2^n - u_2^{n-1}}{\Delta t} = \alpha \frac{u_3^n - 2u_2^n + u_1^n}{\Delta x^2} \tag{12}$$

he boundary values u_0^n and u_3^n are known as zero. Collecting the unknown new alues u_1^n and u_2^n on the left-hand side gives

$$\left(1 + 2\alpha \frac{\Delta t}{\Delta x^2}\right) u_1^n - \alpha \frac{\Delta t}{\Delta x^2} u_2^n = u_1^{n-1}, \tag{13}$$

$$-\alpha \frac{\Delta t}{\Delta x^2} u_1^n + \left(1 + 2\alpha \frac{\Delta t}{\Delta x^2}\right) u_2^n = u_2^{n-1}\,. \tag{14}$$

his is a coupled 2×2 system of algebraic equations for the unknowns u_1^n and $_2^n$. Discretization methods that lead to a coupled system of equations for the nknown function at a new time level are said to be *implicit methods*. The ounterpart, *explicit methods*, refers to discretization methods where there is a mple explicit formula for the values of the unknown function at each of the patial mesh points at the new time level. From an implementational point of iew, implicit methods are more comprehensive to code since they require the olution of coupled equations, i.e., a matrix system, at each time level.

In the general case, (10) gives rise to a coupled $(Nx - 1) \times (Nx - 1)$ system f algebraic equations for all the unknown u_i^n at the interior spatial points $= 1, \ldots, Nx-1$. Collecting the unknowns on the left-hand side, and introducing ne *numerical Fourier number*

$$F_o = \alpha \frac{\Delta t}{\Delta x^2}, \tag{15}$$

.0) can be written

$$-F_o u_{i-1}^n + (1 + 2F_o)\, u_i^n - F_o u_{i+1}^n = u_{i-1}^{n-1}, \tag{16}$$

or $i = 1, \ldots, Nx - 1$. One can either view these equations as a system for where ne u_i^n values at the internal grid points, $i = 1, \ldots, N_x - 1$, are unknown, or we ay append the boundary values u_0^n and $u_{N_x}^n$ to the system. In the latter case, ll u_i^n for $i = 0, \ldots, N_x$ are unknown and we must add the boundary equations o the $N_x - 1$ equations in (16):

$$u_0^n = 0, \tag{17}$$

$$u_{N_x}^n = 0\,. \tag{18}$$

A coupled system of algebraic equations can be written on matr and this is important if we want to call up ready-made software for solv system. The equations (16) and (17)–(18) correspond to the matrix equ

$$AU = b$$

where $U = (u_0^n, \ldots, u_{N_x}^n)$, and the matrix A has the following structure

$$A = \begin{pmatrix}
A_{0,0} & A_{0,1} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \\
A_{1,0} & A_{1,1} & 0 & \ddots & & & & & \\
0 & A_{2,1} & A_{2,2} & A_{2,3} & \ddots & & & & \\
\vdots & \ddots & & \ddots & \ddots & 0 & & & \\
\vdots & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\
\vdots & & & 0 & A_{i,i-1} & A_{i,i} & A_{i,i+1} & \ddots & \\
\vdots & & & & \ddots & \ddots & \ddots & \ddots & \\
\vdots & & & & & \ddots & \ddots & \ddots & A_N \\
0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & A_{N_x,N_x-1} & A
\end{pmatrix}$$

The nonzero elements are given by

$$A_{i,i-1} = -F_o$$
$$A_{i,i} = 1 + 2F_o$$
$$A_{i,i+1} = -F_o$$

for the equations for internal points, $i = 1, \ldots, N_x - 1$. The equations boundary points correspond to

$$A_{0,0} = 1,$$
$$A_{0,1} = 0,$$
$$A_{N_x,N_x-1} = 0,$$
$$A_{N_x,N_x} = 1\,.$$

The right-hand side b is written as

$$b = \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_i \\ \vdots \\ b_{N_x} \end{pmatrix}$$

ith

$$b_0 = 0, \tag{28}$$

$$b_i = u_i^{n-1}, \quad i = 1, \ldots, N_x - 1, \tag{29}$$

$$b_{N_x} = 0\,. \tag{30}$$

We observe that the matrix A contains quantities that do not change in me. Therefore, A can be formed once and for all before we enter the recursive ormulas for the time evolution. The right-hand side b, however, must be updated t each time step. This leads to the following computational algorithm, here ketched with Python code:

```
= linspace(0, L, Nx+1)   # mesh points in space
dx = x[1] - x[0]
= linspace(0, T, N+1)    # mesh points in time
u   = zeros(Nx+1)
u_1 = zeros(Nx+1)

# Data structures for the linear system
A = zeros((Nx+1, Nx+1))
b = zeros(Nx+1)

for i in range(1, Nx):
    A[i,i-1] = -Fo
    A[i,i+1] = -Fo
    A[i,i] = 1 + 2*Fo
A[0,0] = A[Nx,Nx] = 1

# Set initial condition u(x,0) = I(x)
for i in range(0, Nx+1):
    u_1[i] = I(x[i])

import scipy.linalg

for n in range(0, Nt):
    # Compute b and solve linear system
    for i in range(1, Nx):
        b[i] = -u_1[i]
    b[0] = b[Nx] = 0
    u[:] = scipy.linalg.solve(A, b)

    # Update u_1 before next step
    u_1[:] = u
```

.4 Sparse matrix implementation

We have seen from (19) that the matrix A is tridiagonal. The code segment above used a full, dense matrix representation of A, which stores a lot of values e know are zero beforehand, and worse, the solution algorithm computes with ll these zeros. With $N_x + 1$ unknowns, the work by the solution algorithm is $(N_x + 1)^3$ and the storage requirements $(N_x + 1)^2$. By utilizing the fact that A

is tridiagonal and employing corresponding software tools, the work and demands can be proportional to N_x only.

The key idea is to apply a data structure for a tridiagonal or sparse The `scipy.sparse` package has relevant utilities. For example, we can s nonzero diagonals of a matrix. The package also has linear system solv operate on sparse matrix data structures. The code below illustrates can store only the main diagonal and the upper and lower diagonals.

```
# Representation of sparse matrix and right-hand side
main  = zeros(Nx+1)
lower = zeros(Nx-1)
upper = zeros(Nx-1)
b     = zeros(Nx+1)

# Precompute sparse matrix
main[:] = 1 + 2*Fo
lower[:] = -Fo  #1
upper[:] = -Fo  #1
# Insert boundary conditions
main[0] = 1
main[Nx] = 1

A = scipy.sparse.diags(
    diagonals=[main, lower, upper],
    offsets=[0, -1, 1], shape=(Nx+1, Nx+1),
    format='csr')
print A.todense()

# Set initial condition
for i in range(0,Nx+1):
    u_1[i] = I(x[i])

for n in range(0, Nt):
    b = u_1
    b[0] = b[-1] = 0.0  # boundary conditions
    u[:] = scipy.sparse.linalg.spsolve(A, b)
    u_1[:] = u
```

The `scipy.sparse.linalg.spsolve` function utilizes the sparse storag ture of `A` and performs in this case a very efficient Gaussian eliminatio

1.5 The θ rule

The θ rule provides a family of finite difference approximations in time

- $\theta = 0$ gives the Forward Euler scheme in time
- $\theta = 1$ gives the Backward Euler scheme in time
- $\theta = \frac{1}{2}$ gives the Crank-Nicolson scheme in time

Applied to the 1D diffusion problem we have

$$\frac{u_i^{n+1} - u_i^n}{\Delta t} = \alpha \left(\theta \frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} + (1-\theta) \frac{u_{i+1}^n - 2u_i^n + u_{i-}^n}{\Delta x^2} \right.$$

his scheme also leads to a matrix system with entries $1 + 2F_o\theta$ on the main iagonal of the matrix, and $-F_o\theta$ on the super- and sub-diagonal. The right-hand de entry b_i is

$$b_i = u_i^n + F_o(1-\theta)\frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{\Delta x^2}\,.$$

.6 The Laplace and Poisson equation

he Laplace equation, $\nabla^2 u = 0$, or the Poisson equation, $-\nabla^2 u = f$, occur ı numerous applications throughout science and engineering. We can solve D variants of the Laplace equations with the listed software, because we can ıterpret $u_{xx} = 0$ as the limiting solution of $u_t = \alpha u_{xx}$ when u reach a steady :ate limit where $u_t \to 0$. Similarly, Poisson's equation $-u_{xx} = f$ arises from)lving $u_t = u_{xx} + f$ and letting $t \to$ so $u_t \to 0$.

Technically in a program, we can simulate $t \to \infty$ by just taking one large me step, or equivalently, set α to a large value. All we need is to have F_o large. s $F_o \to \infty$, we can from the schemes see that the limiting discrete equation ecomes

$$\frac{u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{\Delta x^2} = 0,$$

hich is nothing but the discretization $[D_x D_x u]_i^{n+1} = 0$ of $u_{xx} = 0$.

The Backward Euler scheme can solve the limit equation directly and hence roduce a solution of the 1D Laplace equation. With the Forward Euler scheme e must do the time stepping since $F_o > 1/2$ is illegal and leads to instability. /e may interpret this time stepping as solving the equation system from u_{xx} by erating on a time pseudo time variable.

.7 Extensions

hese extensions are performed exactly as for a wave equation as they only affect ıe spatial derivatives (which are the same as in the wave equation).

- Variable coefficients
- Neumann and Robin conditions
- 2D and 3D

uture versions of this document will for completeness and independence of the ave equation document feature info on the three points. The Robin condition new, but straightforward to handle:

$$-\alpha\frac{\partial u}{\partial n} = h_T(u - U_s), \quad [-\alpha D_x u = h_T(u - U_s)]_i^n$$

2 Analysis of schemes for the diffusion equ

2.1 Properties of the solution

A particular characteristic of diffusive processes, governed by an equati

$$u_t = \alpha u_{xx},$$

is that the initial shape $u(x,0) = I(x)$ spreads out in space with tim with a decaying amplitude. Three different examples will illustrate the sp of u in space and the decay in time.

Similarity solution. The diffusion equation (31) admits solutions that on $\eta = (x-c)/\sqrt{4\alpha t}$ for a given value of c. One particular solution is

$$u(x,t) = a\,\mathrm{erf}(\eta) + b,$$

where

$$\mathrm{erf}(\eta) = \frac{2}{\sqrt{\pi}}\int_0^\eta e^{-\zeta^2}d\zeta,$$

is the *error function*, and a and b are arbitrary constants. The error f lies in $(-1,1)$, is odd around $\eta = 0$, and goes relatively quickly to ± 1:

$$\begin{aligned}
\lim_{\eta\to-\infty}\mathrm{erf}(\eta) &= -1,\\
\lim_{\eta\to\infty}\mathrm{erf}(\eta) &= 1,\\
\mathrm{erf}(\eta) &= -\mathrm{erf}(-\eta),\\
\mathrm{erf}(0) &= 0,\\
\mathrm{erf}(2) &= 0.99532227,\\
\mathrm{erf}(3) &= 0.99997791\,.
\end{aligned}$$

As $t \to 0$, the error function approaches a step function centered a For a diffusion problem posed on the unit interval $[0,1]$, we may choose at $x = 1/2$ (meaning $c = 1/2$), $a = -1/2$, $b = 1/2$. Then

$$u(x,t) = \frac{1}{2}\left(1 - \mathrm{erf}\left(\frac{x-\frac{1}{2}}{\sqrt{4\alpha t}}\right)\right) = \frac{1}{2}\mathrm{erfc}\left(\frac{x-\frac{1}{2}}{\sqrt{4\alpha t}}\right),$$

where we have introduced the *complementary error function* $\mathrm{erfc}(\eta) = 1$ The solution (34) implies the boundary conditions

$$\begin{aligned}
u(0,t) &= \frac{1}{2}\left(1 - \mathrm{erf}\left(\frac{-1/2}{\sqrt{4\alpha t}}\right)\right),\\
u(1,t) &= \frac{1}{2}\left(1 - \mathrm{erf}\left(\frac{1/2}{\sqrt{4\alpha t}}\right)\right).
\end{aligned}$$

or small enough t, $u(0,t) \approx 1$ and $u(1,t) \approx 1$, but as $t \to \infty$, $u(x,t) \to 1/2$ on $),1]$.

olution for a Gaussian pulse. The standard diffusion equation $u_t = \alpha u_{xx}$ dmits a Gaussian function as solution:

$$u(x,t) = \frac{1}{\sqrt{4\pi\alpha t}} \exp\left(-\frac{(x-c)^2}{4\alpha t}\right). \tag{37}$$

t $t = 0$ this is a Dirac delta function, so for computational purposes one must :art to view the solution at some time $t = t_\epsilon > 0$. Replacing t by $t_\epsilon + t$ in 37) makes it easy to operate with a (new) t that starts at $t = 0$ with an initial ɔndition with a finite width. The important feature of (37) is that the standard eviation σ of a sharp initial Gaussian pulse increases in time according to $= \sqrt{2\alpha t}$, making the pulse diffuse and flatten out.

olution for a sine component. For example, (31) admits a solution of the ɔrm

$$u(x,t) = Qe^{-at}\sin(kx)\ . \tag{38}$$

he parameters Q and k can be freely chosen, while inserting (38) in (31) gives ıe constraint

$$a = -\alpha k^2\,.$$

A very important feature is that the initial shape $I(x) = Q\sin kx$ undergoes damping $\exp(-\alpha k^2 t)$, meaning that rapid oscillations in space, corresponding ɔ large k, are very much faster dampened than slow oscillations in space, ɔrresponding to small k. This feature leads to a smoothing of the initial ɔndition with time.

The following examples illustrates the damping properties of (38). We ɔnsider the specific problem

$$\begin{aligned} u_t &= u_{xx}, \quad x \in (0,1),\ t \in (0,T], \\ u(0,t) &= u(1,t) = 0, \quad t \in (0,T], \\ u(x,0) &= \sin(\pi x) + 0.1\sin(100\pi x)\,. \end{aligned}$$

he initial condition has been chosen such that adding two solutions like (38) ɔnstructs an analytical solution to the problem:

$$u(x,t) = e^{-\pi^2 t}\sin(\pi x) + 0.1e^{-\pi^2 10^4 t}\sin(100\pi x)\,. \tag{39}$$

igure 1 illustrates the rapid damping of rapid oscillations $\sin(100\pi x)$ and the ery much slower damping of the slowly varying $\sin(\pi x)$ term. After about $= 0.5\cdot 10^{-4}$ the rapid oscillations do not have a visible amplitude, while we ave to wait until $t \sim 0.5$ before the amplitude of the long wave $\sin(\pi x)$ becomes ery small.

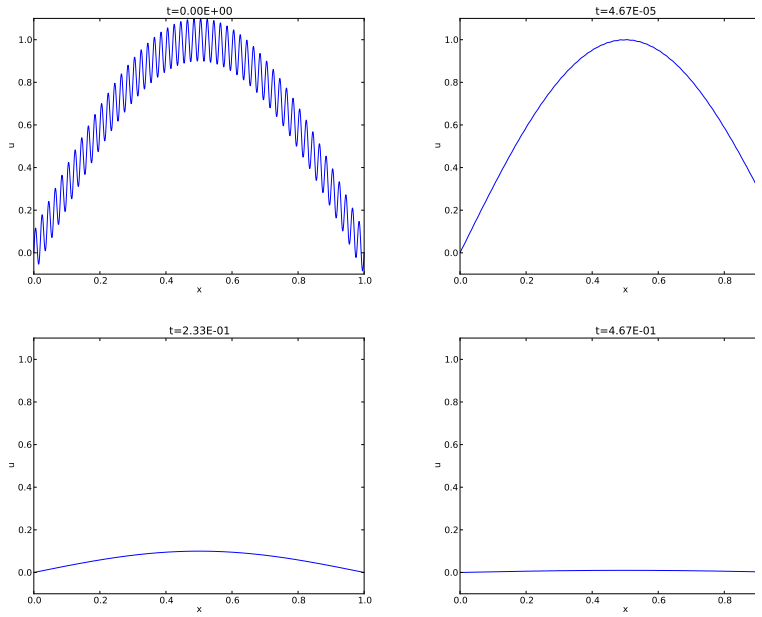


Figure 1: Evolution of the solution of a diffusion problem: initial cc (upper left), 1/100 reduction of the small waves (upper right), 1/10 redu the long wave (lower left), and 1/100 reduction of the long wave (lower

2.2 Analysis of discrete equations

A counterpart to (38) is the complex representation of the same functic

$$u(x,t) = Qe^{-at}e^{ikx},$$

where $i = \sqrt{-1}$ is the imaginary unit. We can add such functions, often to as wave components, to make a Fourier representation of a general sol the diffusion equation:

$$u(x,t) \approx \sum_{k\in K} b_k e^{-\alpha k^2 t}e^{ikx},$$

where K is a set of an infinite number of k values needed to constı solution. In practice, however, the series is truncated and K is a finite values need build a good approximate solution. Note that (39) is a spec of (40) where $K = \{\pi, 100\pi\}$, $b_\pi = 1$, and $b_{100\pi} = 0.1$.

The amplitudes b_k of the individual Fourier waves must be determin the initial condition. At $t = 0$ we have $u \approx \sum_k b_k \exp(ikx)$ and find K such that

$$I(x) \approx \sum_{k \in K} b_k e^{ikx}\,. \tag{41}$$

The relevant formulas for b_k come from Fourier analysis, or equivalently, a least-squares method for approximating $I(x)$ in a function space with basis $\exp(ikx)$.)

Much insight about the behavior of numerical methods can be obtained by investigating how a wave component $\exp(-\alpha k^2 t)\exp(ikx)$ is treated by the numerical scheme. It appears that such wave components are also solutions of the schemes, but the damping factor $\exp(-\alpha k^2 t)$ varies among the schemes. To ease the forthcoming algebra, we write the damping factor as A^n. The exact amplification factor corresponding to A is $A_{\mathrm{e}} = \exp(-\alpha k^2 \Delta t)$.

.3 Analysis of the finite difference schemes

We have seen that a general solution of the diffusion equation can be built as a linear combination of basic components

$$e^{-\alpha k^2 t} e^{ikx}\,.$$

A fundamental question is whether such components are also solutions of the finite difference schemes. This is indeed the case, but the amplitude $\exp(-\alpha k^2 t)$ might be modified (which also happens when solving the ODE counterpart $u' = -\alpha u$). We therefore look for numerical solutions of the form

$$u_q^n = A^n e^{ikq\Delta x} = A^n e^{ikx}, \tag{42}$$

where the amplification factor A must be determined by inserting the component into an actual scheme.

Stability. The exact amplification factor is $A_{\mathrm{e}} = \exp(-\alpha^2 k^2 \Delta t)$. We should therefore require $|A| < 1$ to have a decaying numerical solution as well. If $-1 \leq A < 0$, A^n will change sign from time level to time level, and we get stable, non-physical oscillations in the numerical solutions that are not present in the exact solution.

Accuracy. To determine how accurately a finite difference scheme treats one wave component (42), we see that the basic deviation from the exact solution is reflected in how well A^n approximates A_{e}^n, or how well A approximates A_{e}.

.4 Analysis of the Forward Euler scheme

The Forward Euler finite difference scheme for $u_t = \alpha u_{xx}$ can be written as

$$[D_t^+ u = \alpha D_x D_x u]_q^n\,.$$

Inserting a wave component (42) in the scheme demands calculating the terms

$$e^{ikq\Delta x}[D_t^+ A]^n = e^{ikq\Delta x} A^n \frac{A-1}{\Delta t},$$

and

$$A^n D_x D_x [e^{ikx}]_q = A^n \left(-e^{ikq\Delta x} \frac{4}{\Delta x^2} \sin^2\left(\frac{k\Delta x}{2}\right)\right)\,.$$

Inserting these terms in the discrete equation and dividing by $A^n e^{ikq\Delta x}$

$$\frac{A-1}{\Delta t} = -\alpha \frac{4}{\Delta x^2} \sin^2\left(\frac{k\Delta x}{2}\right),$$

and consequently

$$A = 1 - 4F_o \sin^2\left(\frac{k\Delta x}{2}\right),$$

where

$$F_o = \frac{\alpha \Delta t}{\Delta x^2}$$

is the *numerical Fourier number*. The complete numerical solution is th

$$u_q^n = \left(1 - 4F_o \sin^2\left(\frac{k\Delta x}{2}\right)\right)^n e^{ikq\Delta x}\,.$$

Stability. We easily see that $A \leq 1$. However, the A can be less th which will lead to growth of a numerical wave component. The criterion implies

$$4F_o \sin^2(p/2) \leq 2\,.$$

The worst case is when $\sin^2(p/2) = 1$, so a sufficient criterion for stabil

$$F_o \leq \frac{1}{2},$$

or expressed as a condition on Δt:

$$\Delta t \leq \frac{\Delta x^2}{2\alpha}\,.$$

Note that halving the spatial mesh size, $\Delta x \rightarrow \frac{1}{2}\Delta x$, requires Δt to be by a factor of 1/4. The method hence becomes very expensive for fine meshes.

ccuracy. Since A is expressed in terms of F_o and the parameter we now call $= k\Delta x/2$, we also express A_e by F_o and p:

$$A_\mathrm{e} = \exp\left(-\alpha k^2 \Delta t\right) = \exp\left(-4F_o p^2\right).$$

omputing the Taylor series expansion of A/A_e in terms of F_o can easily be one with aid of `sympy`:

```
def A_exact(Fo, p):
    return exp(-4*Fo*p**2)

def A_FE(Fo, p):
    return 1 - 4*Fo*sin(p)**2

from sympy import *
Fo, p = symbols('Fo p')
A_err_FE = A_FE(Fo, p)/A_exact(Fo, p)
print A_err_FE.series(Fo, 0, 6)
```

he result is

$$\frac{A}{A_\mathrm{e}} = 1 - 4F_o\sin^2 p + 2F_o p^2 - 16F_o^2 p^2\sin^2 p + 8F_o^2 p^4 + \cdots$$

ecalling that $F_o = \alpha\Delta t/\Delta x$, $p = k\Delta x/2$, and that $\sin^2 p \leq 1$, we realize that ne dominating error terms are at most

$$1 - 4\alpha\frac{\Delta t}{\Delta x^2} + \alpha\Delta t - 4\alpha^2\Delta t^2 + \alpha^2\Delta t^2\Delta x^2 + \cdots.$$

.5 Analysis of the Backward Euler scheme

iscretizing $u_t = \alpha u_{xx}$ by a Backward Euler scheme,

$$[D_t^- u = \alpha D_x D_x u]_q^n,$$

nd inserting a wave component (42), leads to calculations similar to those rising from the Forward Euler scheme, but since

$$e^{ikq\Delta x}[D_t^- A]^n = A^n e^{ikq\Delta x}\frac{1 - A^{-1}}{\Delta t},$$

e get

$$\frac{1 - A^{-1}}{\Delta t} = -\alpha\frac{4}{\Delta x^2}\sin^2\left(\frac{k\Delta x}{2}\right),$$

nd then

$$A = \left(1 + 4F_o\sin^2 p\right)^{-1}. \tag{48}$$

he complete numerical solution can be written

$$u_q^n = \left(1 + 4F_o\sin^2 p\right)^{-n} e^{ikq\Delta x}. \tag{49}$$

Stability. We see from (48) that $0 < A < 1$, which means that all nu wave components are stable and non-oscillatory for any $\Delta t > 0$.

2.6 Analysis of the Crank-Nicolson scheme

The Crank-Nicolson scheme can be written as

$$[D_t u = \alpha D_x D_x \overline{u}^x]_q^{n+\frac{1}{2}},$$

or

$$[D_t u]_q^{n+\frac{1}{2}} = \frac{1}{2}\alpha\left([D_x D_x u]_q^n + [D_x D_x u]_q^{n+1}\right).$$

Inserting (42) in the time derivative approximation leads to

$$[D_t A^n e^{ikq\Delta x}]^{n+\frac{1}{2}} = A^{n+\frac{1}{2}} e^{ikq\Delta x}\frac{A^{\frac{1}{2}} - A^{-\frac{1}{2}}}{\Delta t} = A^n e^{ikq\Delta x}\frac{A-1}{\Delta t}.$$

Inserting (42) in the other terms and dividing by $A^n e^{ikq\Delta x}$ gives the re

$$\frac{A-1}{\Delta t} = -\frac{1}{2}\alpha\frac{4}{\Delta x^2}\sin^2\left(\frac{k\Delta x}{2}\right)(1 + A),$$

and after some more algebra,

$$A = \frac{1 - 2F_o\sin^2 p}{1 + 2F_o\sin^2 p}.$$

The exact numerical solution is hence

$$u_q^n = \left(\frac{1 - 2F_o\sin^2 p}{1 + 2F_o\sin^2 p}\right)^n e^{ikp\Delta x}.$$

Stability. The criteria $A > -1$ and $A < 1$ are fulfilled for any $\Delta t > 0$

2.7 Summary of accuracy of amplification factors

We can plot the various amplification factors against $p = k\Delta x/2$ for c choices of the F_o parameter. Figures 2, 3, and 4 show how long an waves are damped by the various schemes compared to the exact damp long as all schemes are stable, the amplification factor is positive, ex Crank-Nicolson when $F_o > 0.5$.

The effect of negative amplification factors is that A^n changes sign f time level to the next, thereby giving rise to oscillations in time in an an of the solution. We see from Figure 2 that for $F_o = 20$, waves with p undergo a damping close to -1, which means that the amplitude does n and that the wave component jumps up and down in time. For $F_o = 2$ a damping of a factor of 0.5 from one time level to the next, which is ve smaller than the exact damping. Short waves will therefore fail to be ef

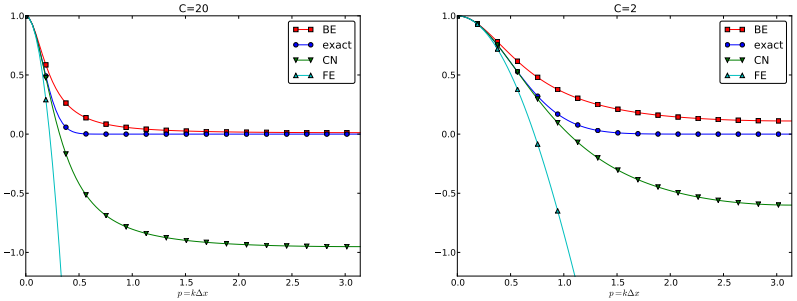
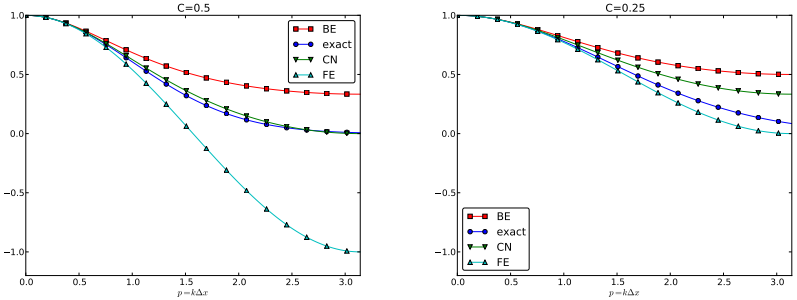


Figure 2: Amplification factors for large time steps.



igure 3: Amplification factors for time steps around the Forward Euler stability mit.

ampened. These waves will manifest themselves as high frequency oscillatory oise in the solution.

A value $p = \pi/4$ corresponds to four mesh points per wave length of e^{ikx}, hile $p = \pi/2$ implies only two points per wave length, which is the smallest umber of points we can have to represent the wave on the mesh.

To demonstrate the oscillatory behavior of the Crank-Nicolson scheme, we oose an initial condition that leads to short waves with significant amplitude. discontinuous $I(x)$ will in particular serve this purpose.

Run $F_o = ...$

xercise 1: Use an analytical solution to formulate a 1D est

his exercise explores the exact solution (37). We shall formulate a diffusion roblem in half of the domain for half of the Gaussian pulse. Then we shall vestigate the impact of using an incorrect boundary condition, which we in eneral cases often are forced due if the solution needs to pass through finite oundaries undisturbed.

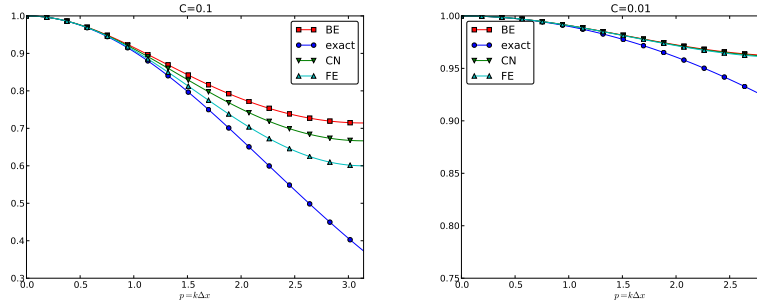


Figure 4: Amplification factors for small time steps.

a) The solution (37) is seen to be symmetric at $x = c$, because $\partial u/$ always vanishes for $x = c$. Use this property to formulate a complet boundary value problem in 1D involving the diffusion equation $u_t = $ $[0, L]$ with $u_x(0, t) = 0$ and $u(L, t)$ known.

b) Use the exact solution to set up a convergence rate test for an implem of the problem. Investigate if a one-sided difference for $u_x(0, t)$, say destroys the second-order accuracy in space.

c) Imagine that we want to solve the problem numerically on $[0, L]$, bu not know the exact solution and cannot of that reason assign a correct I condition at $x = L$. One idea is to simply set $u(L, t) = 0$ since this an accurate approximation before the diffused pulse reaches $x = L$ a thereafter it might be a satisfactory condition. Let u_e be the exact and let u be the solution of $u_t = \alpha u_{xx}$ with an initial Gaussian pulse boundary conditions $u_x(0, t) = u(L, t) = 0$. Derive a diffusion prob the error $e = u_e - u$. Solve this problem numerically using an exact I condition at $x = L$. Animate the evolution of the error and make a curv the error measure

$$E(t) = \sqrt{\frac{\int_0^L e^2 dx}{\int_0^L u dx}}.$$

Is this a suitable error measure for the present problem?

d) Instead of using $u(L, t) = 0$ as approximate boundary condition for let diffused Gaussian pulse out of our finite domain, one may try $u_x(L, t) =$ the solution for large t is quite flat. Argue that this condition gives a co wrong asymptotic solution as $t \to 0$. To do this, integrate the diffusion from 0 to L, integrate u_{xx} by parts (or use Gauss' divergence theorem i arrive at the important property

$$\frac{d}{dt}\int_0^L u(x, t)dx = 0,$$

nplying that $\int_0^L u dx$ must be constant in time, and therefore

$$\int_0^L u(x,t)dx = \int_0^L I(x)dx\,.$$

he integral of the initial pulse is 1.

) Another idea for an artificial boundary condition at $x = L$ is to use a cooling
w

$$-\alpha u_x = q(u - u_S), \tag{52}$$

here q is an unknown heat transfer coefficient and u_S is the surrounding
mperature in the medium outside of $[0, L]$. (Note that arguing that u_S is
pproximately $u(L,t)$ gives the $u_x = 0$ condition from the previous subexercise
at is qualitatively wrong for large t.) Develop a diffusion problem for the error
the solution using (52) as boundary condition. Assume one can take $u_S = 0$
outside the domain" as $u \to 0$ for $x \to \infty$. Find a function $q = q(t)$ such that
e exact solution obeys the condition (52). Test some constant values of q
nd animate how the corresponding error function behaves. Also compute $E(t)$
urves as suggested in subexercise b).
ilename: `diffu_symmetric_gaussian.py`.

Exercise 2: Use an analytical solution to formulate a 2D test

eneralize (37) to multi dimensions by assuming that one-dimensional solutions
an be multiplied to solve $u_t = \alpha\nabla^2 u$. Use this solution to formulate a 2D test
ase where the peak of the Gaussian is at the origin and where the domain is a
ectangule in the first quadrant. Use symmetry boundary conditions $\partial u/\partial n =$
whereever possible, and use exact Dirichlet conditions on the remaining
oundaries. Filename: `diffu_symmetric_gaussian_2D.pdf`.

Exercise 3: Examine stability of a diffusion model with a source term

onsider a diffusion equation with a linear u term:

$$u_t = \alpha u_{xx} + \beta u\,.$$

) Derive in detail a Forward Euler scheme, a Backward Euler scheme, and a
rank-Nicolson for this type of diffusion model. Thereafter, formulate a θ-rule
o summarize the three schemes.

) Assume a solution like (38) and find the relation between a, k, α, and β.

) Calculate the stability of the Forward Euler scheme. Design numerical
xperiments to confirm the results.

d) Repeat c) for the Backward Euler scheme.

e) Repeat c) for the Crank-Nicolson scheme.

f) How does the extra term bu impact the accuracy of the three schem

Hint. Compare the numerical and exact amplification factors, either ir
or by Taylor series expansion (or both).
Filename: `diffu_stab_uterm.pdf`.

3 Exercises

Exercise 4: Stabilizing the Crank-Nicolson method by nacher time stepping

It is well known that the Crank-Nicolson method may give rise to non-
oscillations in the solution of diffusion equations if the initial data exhibi
(see Section 2.6). Rannacher [1] suggested a stabilizing technique consi
using the Backward Euler scheme for the first two time steps with step
$\frac{1}{2}\Delta t$. One can generalize this idea to taking $2m$ time steps of size $\frac{1}{2}\Delta t$ v
Backward Euler method and then continuing with the Crank-Nicolson
which is of second-order in time. The idea is that the high frequencie
initial solution are quickly damped out, and the Backward Euler schem
these high frequencies correctly. Thereafter, the high frequency conten
solution is gone and the Crank-Nicolson method will do well.

Test this idea for $m = 1, 2, 3$ on a diffusion problem with a discon
initial condition. Measure the convergence rate using the solution (34) v
boundary conditions (35)-(36) for t values such that the conditions ar
vicinity of ± 1. For example, $t < 5a1.6 \cdot 10^{-2}$ makes the solution diffusio
step to almost a straight line. The program `diffu_erf_sol.py` shows
compute the analytical solution.

Project 5: Energy estimates for diffusion problems

This project concerns so-called *energy estimates* for diffusion problems t
be used for qualitative analytical insight and for verification of impleme

a) We start with a 1D homogeneous diffusion equation with zero D
conditions:

$$\begin{aligned} u_t &= \alpha u_x x, && x \in \Omega = (0, L),\ t \in (0, T], \\ u(0,t) = u(L,t) &= 0, && t \in (0, T], \\ u(x,0) &= I(x), && x \in [0, L]\,. \end{aligned}$$

The energy estimate for this problem reads

$$||u||_{L^2} \leq ||I||_{L^2}, \tag{56}$$

here the $||\cdot||_{L^2}$ norm is defined by

$$||g||_{L^2} = \sqrt{\int_0^L g^2 dx}\,. \tag{57}$$

he quantify $||u||_{L^2}$ or $\frac{1}{2}||u||_{L^2}$ is known as the *energy* of the solution, although it not the physical energy of the system. A mathematical tradition has introduced ıe notion *energy* in this context.

The estimate (56) says that the "size of u" never exceeds that of the initial ɔndition, or more equivalently, that the area under the u curve decreases with me.

To show (56), multiply the PDE by u and integrate from 0 to L. Use that u_t can be expressed as the time derivative of u^2 and that u_xxu can integrated y parts to form an integrand u_x^2. Show that the time derivative of $||u||_{L^2}^2$ must e less than or equal to zero. Integrate this expression and derive (56).

) Now we address a slightly different problem,

$$u_t = \alpha u_x x + f(x,t), \qquad x \in \Omega = (0,L),\ t \in (0,T], \tag{58}$$
$$u(0,t) = u(L,t) = 0, \qquad t \in (0,T], \tag{59}$$
$$u(x,0) = 0, \qquad x \in [0,L]\,. \tag{60}$$

he associated energy estimate is

$$||u||_{L^2} \leq ||f||_{L^2}\,. \tag{61}$$

This result is more difficult to derive.)

Now consider the compound problem with an initial condition $I(x)$ and a ght-hand side $f(x,t)$:

$$u_t = \alpha u_x x + f(x,t), \qquad x \in \Omega = (0,L),\ t \in (0,T], \tag{62}$$
$$u(0,t) = u(L,t) = 0, \qquad t \in (0,T], \tag{63}$$
$$u(x,0) = I(x), \qquad x \in [0,L]\,. \tag{64}$$

how that if w_1 fulfills (53)-(55) and w_2 fulfills (58)-(60), then $u = w_1 + w_2$ is ıe solution of (62)-(64). Using the triangle inequality for norms,

$$||a+b|| \leq ||a|| + ||b||,$$

ıow that the energy estimate for (62)-(64) becomes

$$||u||_{L^2} \leq ||I||_{L^2} + ||f||_{L^2}\,. \tag{65}$$

c) One application of (65) is to prove uniqueness of the solution. Sup and u_2 both fulfill (62)-(64). Show that $u = u_1 - u_2$ then fulfills (62)-(6 $f = 0$ and $I = 0$. Use (65) to deduce that the energy must be zero for a and therefore that $u_1 = u_2$, which proves that the solution is unique.

d) Generalize (65) to a 2D/3D diffusion equation $u_t = \nabla\cdot(\alpha\nabla u)$ for x

Hint. Use integration by parts in multi dimensions:

$$\int_\Omega u\nabla\cdot(\alpha\nabla u)\,\mathrm{d}x = -\int_\Omega \alpha\nabla u\cdot\nabla u\,\mathrm{d}x + \int_{\partial\Omega} u\alpha\frac{\partial u}{\partial n},$$

where $\frac{\partial u}{\partial n} = \boldsymbol{n}\cdot\nabla u$, $\boldsymbol{n}$ being the outward unit normal to the boundary ∂ domain Ω.

e) Now we also consider the multi-dimensional PDE $u_t = \nabla\cdot(\alpha\nabla u)$. I both sides over Ω and use Gauss' divergence theorem, $\int_\Omega \nabla\cdot\boldsymbol{q}\,\mathrm{d}x = \int_{\partial\Omega}$ for a vector field $\boldsymbol{q}$. Show that if we have homogeneous Neumann condi the boundary, $\partial u/\partial n = 0$, area under the u surface remains constant and

$$\int_\Omega u\,\mathrm{d}x = \int_\Omega I\,\mathrm{d}x\,.$$

f) Establish a code in 1D, 2D, or 3D that can solve a diffusion equatio source term f, initial condition I, and zero Dirichlet or Neumann condi the whole boundary.

We can use (65) and (66) as a partial verification of the code. Choo functions f and I and check that (65) is obeyed at any time when zero I conditions are used. Iterate over the same I functions and check that fulfilled when using zero Neumann conditions.

g) Make a list of some possible bugs in the code, such as indexing e arrays, failure to set the correct boundary conditions, evaluation of a te wrong time level, and similar. For each of the bugs, see if the verificati from the previous subexercise pass or fail. This investigation shows hov the energy estimates and the estimate (66) are for pointing out error implementation.
Filename: `diffu_energy.pdf`.

References

[1] R. Rannacher. Finite element solution of diffusion problems with i data. *Numerische Mathematik*, 43:309–327, 1984.

ndex